Universidad Autónoma de Madrid
Escuela Politécnica Superior

Proyecto fin de grado

PLANIFICACIÓN DE MISIONES PARA VEHÍCULOS NO TRIPULADOS (UAS)

Grado en Ingeniería Informática

AUTOR: Jaime Fradera Gil
TUTOR: Gema Bello
PONENTE: David Camacho
21 de julio de 2014

PLANIFICACIÓN DE MISIONES PARA VEHÍCULOS NO TRIPULADOS (UAS)

AUTOR: Jaime Fradera Gil
TUTOR: Gema Bello
PONENTE: David Camacho

Dpto. de Ingeniera Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
21 de julio de 2014

Resumen

Resumen

Los primeros vehículos aéreos no tripulados aparecieron en la década de los 70, pero debido a su alto coste y su escaso desarrollo no ofrecían suficientes ventajas respecto de los servicios que podía dar un piloto experimentado en una aeronave tripulada de forma tradicional. Es en estos últimos años cuando el uso de este tipo de vehículos está comenzando a surgir en una gran cantidad de ámbitos. Algunos son la observación y análisis de terrenos o tareas de vigilancia y monitorización.

Actualmente la problemática de la planificación de misiones para UAVs es el hecho de tener que ser realizada a mano por un humano. Por este motivo, se han desarrollado diversos sistemas para la automatización de este tipo de tareas.

Los planificadores actuales basados en búsquedas en grafo o mediante motores de inferencia lógica tienen una serie de limitaciones, una de las más importantes es el coste computacional. Usualmente las misiones generadas necesitan cumplir una gran cantidad de requisitos, además de tener que coordinar diferentes vehículos, lo que genera árboles de búsqueda que necesitan una gran capacidad de procesamiento para poder encontrar una solución.

Dado que la planificación de misiones se basa en problemas de búsqueda, existe otra opción para resolver este tipo de problemas: los algoritmos genéticos. Estos algoritmos siguen la misma filosofía que la selección natural de Darwin, es decir, la conservación de los individuos mejor adaptados al problema planteado. Al seguir esta teoría, no es necesario realizar una búsqueda completa de todas las posibilidades, consiguiendo así un rendimiento mayor a los algoritmos clásicos de búsqueda.

Por este motivo, en este trabajo se ha decidido implementar un algoritmo genético para resolver el problema de la planificación automática de misiones con múltiples UAVs. Para ello, se ha seleccionado el modelo simple de los algoritmos genéticos, basado en las cuatro etapas principales de estos algoritmos. Para comprobar el correcto funcionamiento del sistema se ha realizado dos experimentos para hallar la solución óptima al problema y además estudiar el tiempo de cómputo necesario para alcanzar dicha solución.

Palabras Clave

Algoritmo genético, Fitness, Individuo, Misión, Planificación de misiones, Vehículos aéreos no tripulados.

Abstract

The first unmanned air vehicles appeared in 70's, but at the beginning they had a very high cost and they did not offer enough advantages compared with an experienced pilot using a traditional aircraft. In the last years, UAVs are being used in many diferents scopes. Usually this missions are surveillance tasks, detecting forest fire, rescue tasks, delivery packets or just entertainment.

Today, the main problem for planning missions with UAVs is the fact that this task is needed be done by a person and it is totally performed manually. For this reason, it has been developed many plannifiers for the automation of these tasks.

Nowadays, existing planners are based on graph search or use a logic engine. But this kind of planners have several limitations, probably the most important is the big computational cost that this algorithm needs to resolve this missions. These missions have a lot of requirements that need to be considered and also it is necessary to coordinate all the UAV collection. These requirementes generate search graphs that need a huge process' capabilities to find a solution.

Due to mission planning is based on search problems, there is another option to solve this type of problems: genetic algorithms. These algorithms are inspired by the natural selection of Darwin, thus only the fittest individuals survive. So, we do not need to do a full search, improving the througput of our system.

But we have selected other type of algorithm that can avoid this problem, we have decided to use genetic algorithms for develop a mission planner that can resolve missions with a big amount of tasks.

For this reason, we have decided to implement a genetic algorithm to solve the problem of automatic mission planning with UAVs. To achieve that, a simple model of the genetic algorithm has been used. This kind of algorith is based on for stages. To prove the proper behaviour of the system, two experiments has been done to find an optimal solution and in addition, studying the computation time needed to reach the solution.

Key words

UAV, Genetic algorithm, Fitness, Individual, Mission, Mission planner.

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción

En este capítulo desarrollamos las motivaciones que han inspirado este proyecto, y los objetivos que con este se desean conseguir. Además de proporcionar un esquema de la estructura del proyecto que aquí se presenta.

1.1. Motivación y definición del problema

Las posibles aplicaciones de la UAS (Unmanned Air System) son muchas y variadas, como pueden ser la vigilancia de territorios o el tráfico por carretera, rescates, acceder a localizaciones con alto riesgo, la agricultura y la inspección de territorios, etc. Durante los últimos 20 años, un gran número de trabajos de investigación en este ámbito se han llevado a cabo, aplicando diversas técnicas y algoritmos para aumentar las capacidades autónomas de estos sistemas.

El sistema de guía dentro de los UAS se podría definir como el conductor, ejerciendo las funciones de planificación y de toma de decisiones para lograr que las misiones y objetivos que se tengan asignados sean alcanzados. La función de este subsistema es reemplazar los procesos cognitivos de un piloto humano u operador.

Este trabajo trata de abordar una de las partes fundamentales de este subsistema de guía que es la planificación de misiones mediante el uso de algoritmos genéticos. Dado que este tipo de algoritmos ofrecen un gran rendimiento computacional, son idóneos para la optimización de problemas que requieren gran capacidad de cómputo como es el caso de la planificación de misiones.

1.2. Objetivos del proyecto

En este proyecto se va a plantear una solución al problema actual sobre la planificación y optimización de misiones para vehículos aéreos no tripulados mediante algoritmos genéticos.

Con esto se pretende mejorar la actual tendencia a usar algoritmos de planificación centrados en búsqueda en grafo, que a pesar de ofrecer soluciones en algunos casos óptimas para la misión planteada, están limitadas por el coste computacional que conlleva este tipo de algoritmos,

especialmente para misiones complejas que requieran de coordinación entre vehículos, o posean un gran número de tareas.

Por ello, se ha optado por utilizar algoritmos genéticos como método de resolución debido a que, con un coste computacional mucho menor son capaces de obtener soluciones potencialmente óptimas para misiones complejas. Para el desarrollo de este proyecto se ha desarrollado un modelo de datos para emular un sistema de UAVs, a los que se les podrá asignar tareas que formarán una misión completa.

El sistema esta desarrollado de manera que cualquier solución obtenida mediante el algoritmo sea capaz de resolver el 100 % de las tareas dadas, quedando descartada cualquier misión que posea tareas no resolubles por el conjunto de UAVs dados.

Finalmente, se estudiarán los resultados obtenidos con el sistema, con el fin de aportar unos resultados que muestran cómo se comporta el algoritmo genético diseñado según aumenta la complejidad de las misiones a resolver.

1.3. Estructura del documento

En esta sección se da una breve descripción de del contenido que se puede encontrar en cada capítulo de este documento. Este documento está compuesto por los siguientes siete capítulos:

- **Introducción**: En este capítulo se ofrece una introducción al trabajo realizado, y a las motivaciones y objetivos que ha propiciado la realización de este.
- **Estado del arte**: En este capítulo se muestra una visión global de estado actual de los algoritmos genéticos, la planificación de misiones, y los UAV.
- **Modelo del escenario**: En este capítulo se muestra el modelo diseñado para recrear los escenarios de las misiones que se deben planificar, y los elementos que en ellas intervienen.
- **Algoritmos genéticos para planificación**: En este capítulo se desarrolla el algoritmo genético que se ha codificado para realizar la planificación.
- **Experimentación y resultados obtenidos**: En este capítulo se muestran resultados reales de pruebas realizadas con misiones establecidas en el sistema.
- **Conclusiones**: En este capítulo se muestran las conclusiones que se han obtenido tras la realización de este proyecto.
- **Posibles mejoras**: En este capítulo se muestra un listado de las posibles mejoras que se podrían añadir en un futuro al sistema desarrollado.

Capítulo 2

Estado del arte

En este capítulo se comentarán los aspectos actuales sobre los algoritmos genéticos, vehículos aéreos no tripulados y la planificación de misiones. Esto se ha realizado para facilitar la correcta compresión del resto del trabajo.

2.1. Algoritmos genéticos

En esta sección se explican los aspectos más relevantes de los algoritmos genéticos.

2.1.1. Introducción a los AG

Los algoritmos genéticos se diseñaron alrededor de 1970 como una de las líneas mas importantes y prometedoras de la inteligencia artificial como método de resolución de problemas (especialmente relacionados con la optimización de búsqueda de soluciones). Estos algoritmos reciben este nombre debido a la influencia directa que reciben del proceso de evolución biológica o selección natural [7].

Este tipo de algoritmos se basan en la generación inicial de individuos, usualmente partiendo de una semilla aleatoria (determinados por su genotipo según las necesidades del problema que se esta representando) a los que se les aplica una serie de acciones u *operadores*, del mismo modo que ocurre en la evolución biológica con mutaciones y recombinaciones genéticas. A modo de ejemplo, a continuación se explican brevemente algunos de los operadores con los que actúa un algoritmo genético simple.

Selección: Este operador se encarga de seleccionar los individuos mejor preparados según la puntuación obtenida en la función que evalúa cómo de buena es esa aproximación a la solución óptima. Esta selección puede realizarse de diversas maneras, aunque usualmente y para este tipo de algoritmos simples existen dos modos a proceder predominantes, el primero trata de seleccionar únicamente los mejores N individuos para su posterior cruce, mientras que el segundo método selecciona los M mejores individuos promocionándolos sin sufrir ninguna modificación a la siguiente generación, y finalmente para completar el resto de población se utilizan de nuevo los M mejores individuos para realizar cruces que generen nuevos individuos.

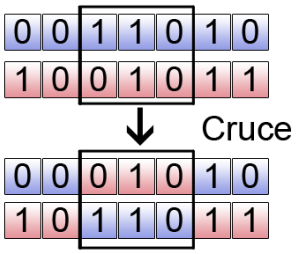

Figura 2.1: Cruce simple en dos puntos binario

Tras cada generación se deberán evaluar los nuevos individuos generados para medir como se acercan a la solución óptima del problema. Este proceso se realiza en cada una de las generaciones mediante una función de ajuste o *fitness*, de tal manera que todos quedan clasificados mediante el mismo criterio.

Una vez calculada la función fitness de cada individuo existen diferentes alternativas para seleccionar los padres que se van a utilizar en la fase de cruce. Una de las mas comunes consiste generar un torneo entre los individuos mejor clasificados según la puntuación obtenida tras la evaluación de la función fitness, generando una nueva colección de individuos **nuevos y no repetidos** hasta completar el total de individuos de la población especificada en el problema (cabe destacar que el número de individuos de una población puede ser variable según los criterios del problema).

Por otro lado, al anterior método citado se puede añadir un método de **selección elitista** de N individuos, es decir, se promocionan para la siguiente generación a los N mejores clasificados según la fitness utilizada, sin sufrir ningún tipo de modificación, mientras que el resto de nuevos individuos surgirá de nuevo mediante cruces entre los mejores clasificados tras la anterior evaluación. [8]

Cruce: Este operador se encarga de generar nuevos individuos, realizando combinaciones de los N mejores candidatos seleccionados previamente construyendo así la siguiente población. El cruce de estos individuos depende exclusivamente del modelo de individuo y de problema que se esté tratando. A modo de ejemplo, en la Figura 2.1 se muestra el cruce entre dos individuos con genotipo binario.

Mutación: Este operador tiene la función de incrementar la variación en la diversidad de los individuos, evitando así una caída en óptimos locales. Con este operador se consigue evitar la convergencia temprana entre los individuos, favoreciendo así que la solución use una muestra de poblaciones mayor y, por tanto, dicha solución sea más próxima a la solución óptima del problema. Usualmente la mutación debe tener una probabilidad baja, ya que la utilización de valores muy altos en este operador tiende invertir el efecto deseado, convergiendo cada vez a resultados peores debido a su carácter aleatorio.

Como podemos observar, en el *Algoritmo 1* se muestra el pseudocódigo para un algoritmo genético simple con los pasos que se han citado previamente en este apartado. En este caso, el tipo de selección no es elitista, ya que no se producen copias de individuos. Además, a partir de dos individuos ganadores se generan otros dos individuos como se explica en la Figura 2.1, para después aplicarles una mutación según el parámetro *mutProb*.

Algoritmo 1: Pseucodcódigo de algoritmo genético simple

Output: El individuo $I_i = \{g_1, g_2, \ldots, g_n\}$ cuya $Fitness(I_i)$ esta maximizada hasta 1

$I \leftarrow$ Conjunto aleatorio de individuos de la población
$B \leftarrow$ Conjunto de individuos ordenados según su evaluación de $Fitness$
λ Número de mejores individuos que se seleccionarán para realiar cruces
$i \leftarrow 1$
$convergence \leftarrow 0$
while $i \leq generaciones \wedge convergencia = 0$ **do**
    $B \leftarrow \emptyset$
    for $j \leftarrow 1$ ***to*** *poblacion* **do**
        $B \leftarrow Fitness(I_j)$
    $Cbest \leftarrow SeleccionNMejores(\lambda, B)$
    $C \leftarrow Cbest$
    for $j \leftarrow$ *1* ***to*** λ **do**
        $p1, p2 \leftarrow$ selección de pares de los mejores individuos de $Imej$
        $i1 \leftarrow Cruce(p1, p2, 1)$
        $i2 \leftarrow Cruce(p1, p2, 2)$
        $i1 \leftarrow mutacion(i1, mutProb)$
        $i2 \leftarrow mutacion(i2, mutProb)$
        $C \leftarrow I \cup \{i1, i2\}$
    $i \leftarrow i + 1$
    $convergencia \leftarrow compruebaConvergencia(Cbest)$
return *SeleccionMejor(C,F)*

2.1.2. Usos frecuentes

Desde la aparición de los algoritmos genéticos han sido muchos los ámbitos, especialmente en ingeniería e investigación, donde han sido empleados debido a su potencia y eficacia. Algunos de estos son:

Optimización: Este ámbito es uno de los más relevantes en el uso de algoritmos genéticos. Esto se debe al gran coste computacional y de tiempo que suelen requerir muchos algoritmos, mientras que los algoritmos genéticos consiguen resultados potencialmente óptimos en tiempos mucho menores, gracias a que no necesitan explorar todas las posibilidades para obtener una solución óptima. Por ejemplo, existen proyectos para la optimización de transmisiones de engranajes, consiguiendo no solo el diseño óptimo para este tipo de elementos, sino también diseños cercanos que puedan cumplir con el diseño establecido [12].

Programación automática: Este tipo de algoritmos ha tenido gran repercusión en el ámbito de la programación debido a la implementación de estos para generar y optimizar programas mediante técnicas evolutivas. Uno de los ejemplos mas representativos en este área es la programación automática de robots, la cual consigue que robots diseñados mediante este algoritmo puedan reprogramar su base para aprender a realizar tareas de forma más eficiente [5].

Aprendizaje automático: Los algoritmos genéticos también son utilizados para realizar tareas que requieren de un entrenamiento previo para producir predicciones futuras, como

podría ser la predicción del tiempo atmosférico o la predicción de estructuras protéicas. En este apartado también cabe destacar el aprendizaje orientado a la robótica mediante la mejora de los sensores por medio de algoritmos genéticos. Un ejemplo común de la utilización de algoritmos genéticos para el el aprendizaje automático es el reconocimiento de patrones [9].

Modelos de sistemas sociales: La utilización de algoritmos genéticos también ha repercutido en el estudio de sistemas sociales, consiguiendo predicciones de comportamiento en grandes poblaciones, cómo por ejemplo, predicciones en la evolución y cooperación de estas [8].

2.2. UAV (Unmanned Air Vehicles)

Actualmente existen tres tipos de vehículos que son capaces de volar sin ningún piloto en su interior, estos se clasifican en **UAV** (vehículos aéreos no tripulados, VANT en castellano), **RPV** (Vehículos pilotados remotamente) y, finalmente, **drones**. Esta última nomenclatura es la que mayor aceptación y alcance ha tenido desde sus inicios, la cual simplemente se refiere a cualquier vehículo ya sea aéreo, terrestre o marítimo capaz de ser pilotado remotamente o por una programación previa mediante sistemas automáticos de navegación.

Por ello, para diferenciar entre estos han aparecido los términos RPV, para vehículos pilotados por un agente humano por medio de radiocontrol, y UAV, vehículos capaces de ser pilotados de forma automática, mediante sensores y sistemas de navegación GPS [10].

Este tipo de vehículos se puede clasificar dependiendo de las tareas para los que son requeridos:

- **Reconocimiento**: Este tipo se utiliza principalmente para la recolección de información, como puede ser el análisis de zonas forestales para detección de incendios o control de terrenos en agricultura.
- **Apoyo**: Destinados a misiones peligrosas como apoyo o misiones donde no se desea poner en riesgo vidas humanas.
- **Logístico**: Estos UAVs tienen la función principal de transporte de carga. Principalmente dedicados a tareas de rescate o recogida de elementos en zonas de difícil acceso que podrían poner en peligro vidas humanas.
- **Civiles**: Utilizados principalmente como ocio y entretenimiento.

Debido a la gran diversidad de UAVs, existen grandes diferencias en las características y capacidades que pueden tener. Por ejemplo en la Figura 2.2 podemos apreciar dos tipos muy distintos de UAVs. El primero de ellos es un micro UAV de tipo *handheld* con un alcance de 3Km y una autonomía de 30 minutos, con una envergadura de 67cm. Por otro lado tenemos UAVs capaces de recorrer largas distancias (hasta 700 Km) y unas alturas de 20000 metros como es la categoría de los UAV *Orbital*.

Esta gran variedad en las características de los UAVs los hace capaces de cumplir una gran cantidad de tareas diferentes, es lo que ha hecho que los UAVs cada vez sean un recurso más a tener en cuenta.

Figura 2.2: Micro UAV Spy Arrow vs Orbital Global Observer

2.2.1. Usos frecuentes

En estos últimos años cada vez se están integrando los UAVs para realizar tareas más complejas, gracias al equipamiento del que disponen como sensores infrarrojos, cámaras de alta definición, grandes capacidades de carga, radar, etc. Algunas tareas que actualmente se realizan mediante UAVs en diferentes ámbitos son las siguientes:

- Cartografía y modelado de terrenos
- Agricultura: control de plagas y monitorización de cultivos
- Medio ambiente: control del estado atmosférico
- Seguridad y vigilancia
- Seguimiento, prevención y extinción de incendios
- Tareas de rescate y salvamento

Algunas de estas tareas cada vez cobran más importancia como, por ejemplo la prevención y extinción de incendios o las tareas de rescate y salvamento, ya que este tipo de tareas entrañan un alto riesgo para el piloto del vehículo, mientras que mediante el uso de UAVs no se pone en peligro ninguna vida humana. Además, la ausencia de un piloto hace que los vehículos puedan ser más pequeños y maniobrables, por lo que pueden acceder a zonas a las que un vehículo estándar no podría.

Por ejemplo, en un artículo publicado por National Geographic se muestran algunos de los usos de los drones en el medio ambiente. Estos usos son completamente diferentes entre ellos, por ejemplo, actualmente tanto la NASA como la NOAA (National Oceanic and Atmospheric Administration) utilizan UAVs de largo alcance para investigar huracanes y grandes tormentas con el fin de poder predecir nuevos fenómenos atmosféricos en un futuro.

Por otro lado, y en este caso los UAVs utilizados no son más grandes que una maqueta de aeromodelismo, están siendo utilizados para crear mapas en 3D mediante la captura de imágenes en alta definición [4].

2.3. Planificación de misiones

Debido a la rápida evolución de las capacidades de los UAVs, estos están siendo incorporados en multitud de ámbitos para realizar tareas cada vez más complejas. A causa de este incremento en la complejidad en las misiones que realizan, ha surgido la necesidad de automatizar la planificación de las tareas que componen la misión, teniendo principalmente en cuenta factores como el ahorro de costes y optimización del tiempo de uso de los UAVs, además de resolver el problema de cumplir tareas de forma simultánea mediante diferentes UAVs.

Por estos motivos, se ha desarrollado lo que se conoce como **planificación de misiones** (mission planning) con el fin de poder automatizar la mayor parte de la coordinación entre UAVs, de tal manera que el control de la misión y de las tareas pase a un modo de gestión automático y el agente humano solo necesite realizar tareas de selección de misión abstrayendose del control individual de UAVs o tareas concretas.

Una fase posterior a la planificación de misiones sería el *path planning*, es decir, la búsqueda de las rutas óptimas para llegar a los objetivos de las tareas que se deben realizar. En estos momentos, este proceso esta altamente automatizado debido a que los UAVs están dotados de sistemas de posicionamiento GPS, además de incorporar sensores para la detección de obstáculos y control guiado [11].

El path planning usualmente esta basado en algoritmos de búsqueda en grafo, con el que se busca el camino mas corto, o de menor coste para alcanzar todas las tareas. Pero esto tiene una limitación computacional ya que este tipo de algoritmos suelen tener un gran coste de computo y se vuelven inviables si el número de tareas aumenta en número o en complejidad.

En la Figura 2.3 se muestra un ejemplo de lo que sería una ruta obtenida a partir de un algoritmo de path planning con obstáculos y áreas restringidas para el UAV que realiza la tarea [3].

Una vez resuelto el problema de la planificación de rutas, se abre un nuevo nivel de complejidad con el problema planteado por la **planificación de misiones**, donde se debe buscar una solución para resolver una misión compuesta por un número de tareas que serán asignadas a la colección de UAVs disponible.

Con la planificación de misiones se alcanza un nuevo nivel de complejidad para el uso de UAVs, puesto que engloba tanto el problema de la resolución de las tareas como la coordinación en paralelo de todo el conjunto de UAVs.

Como se ha explicado anteriormente, el problema de la planificación de rutas ya ha alcanzado un alto grado de automatización y son los propios vehículos, gracias a su equipamiento, los que realizan esta tarea. En cambio, esto es diferente en la planificación de misiones donde actualmente queda en manos de un agente humano la selección y asignación de tareas.

En la figura 2.4 se muestra un ejemplo de una misión realizada mediante planificación de misiones por un UAV único.

Esto genera un problema cada vez mas complejo, ya que según aumenta el número de tareas a realizar o el número de UAVs a coordinar, la dificultad de encontrar de manera manual una misión óptima o incluso una solución. Esto es lo que ha provocado la generación de algoritmos que puedan resolver dicho problema. Algunos de los planificadores que han sido desarrollados históricamente se basan en problemas de búsqueda en grafo, algunos de estos algoritmos están

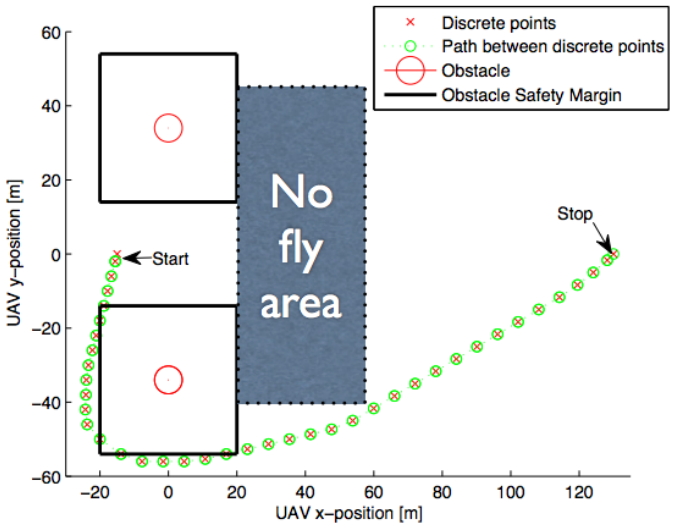

Figura 2.3: Ejemplo de planificación de ruta

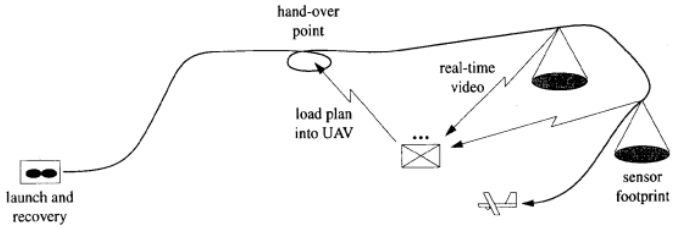

Figura 2.4: Ejemplo de planificación de misiones

explicados en la sección 2.3.1.

Debido al aumento de complejidad, puede resultar que el cómputo de las misiones sea inviable con los planificadores tradicionales, puesto que como ya se ha mencionado, disparan su tiempo de cálculo cuanto más compleja se vuelve la misión a resolver.

Debido a esto se han desarrollado otras técnicas para la resolución y óptimiacion de estas misiones como los **COP (Constraint Optimization Problem)** mediante un motor de resolución de **STRIPS**, y por último, mediante **Algoritmos genéticos** como se explica en la sección 2.3.2.

2.3.1. Planificadores basados en búsqueda en grafo

Existen diferentes formas de planificación de misiones, algunos de los algoritmos más utilizados son lo que se citan a continuación.

Los algoritmos basados en búsqueda hacia delante se basan en búsquedas en árbol donde cada nodo es un estado de la misión y el coste de la misión viene dado mediante las ramas del árbol [6, p. 35-38].

- **Búsqueda en anchura**: Crecimiento uniforme con un rendimiento $O(|V|+|E|)$ donde **V** es el número de vértices y **E** el número de aristas. Este tipo de búsqueda favorece misiones de **pocas tareas**, pero si existe alguna solución se garantiza que la primera encontrada será la de menor coste.
- **Búsqueda en profundidad**: Crecimiento no uniforme con un rendimiento $O(|V|+|E|)$ donde **V** son el número de vértices, y **E** el número de aristas. Este tipo de búsqueda favorece misiones de **muchas tareas**.
- **Algoritmo de Dijkstra**: Encuentra soluciones de coste óptimo con un rendimiento $O(|V|lg|V|+|E|)$. Ofrece el mismo rendimiento independientemente del número de tareas.
- **Algoritmo A***: Es una extensión del algoritmo de Dijkstra que reduce la cantidad de estados visitados con una heurística. Garantiza una solución de coste óptimo explorando menos nodos que Dijkstra.

2.3.2. Otras técnicas de planificación

Por otro lado, existen otro tipo de algoritmos que han sido desarrollados más recientemente con el objetivo de mejorar los algoritmos basados en búsqueda ya que, todos ellos poseen una limitación debido a su alto coste computacional. Algunos de estos algoritmos son los siguientes:

STRIPS (Stanford Research Institute Problem Solver)

Este algoritmo fue creado en la universidad de Stanford. Se basa en un motor de resolución basado en lógica de predicados, diseñado específicamente para la planificación automática [2]. Para resolver un problema mediante STRIPS se necesitan una serie de requisitos:

- **Estado inicial**: Es el punto de partida de la misión.
- **Estado final**: Es el estado que el planificador debe alcanzar para finalizar la ejecución con una solución.
- **Conjunto de acciones**: Es el conjunto de acciones que el motor puede realizar según el estado en que se encuentre.
  - Precondiciones: Cada acción posee condiciones que deben cumplirse previa realización de la acción.
  - Postcondiciones: Son las condiciones que se generan por el hecho de haber realizado la acción.

Además cabe destacar que la planificación mediante STRIPS puede realizarse de forma conjunta con algoritmos como el A* consiguiendo así costes óptimos tambien para este método [6, p. 58].

CSP (Constraint Satisfaction Problem)

Este tipo de planificadores está basado en problemas matemáticos donde el estado de un conjunto de objetos debe satisfacer un número de restricciones.

Este tipo de planificación plantea un inconveniente a la hora de realizar planificación de misiones, debido a que requiere de una búsqueda combinatoria para la resolución del problema, lo que puede desembocar en grandes costes computacionales para misiones complejas, o con gran cantidad de variables ya sea por la necesidad de coordinar múltiples vehículos, o por la necesidad de resolver misiones basadas en una gran cantidad de tareas [1].

2.3.3. Optimización para planificadores: Algoritmos genéticos

La planificación basada en algoritmos genéticos, se plantea como otra alternativa de planificación que intenta paliar el problema común de la mayoría de los planificadores, el coste computacional para la búsqueda de soluciones.

Este tipo de algoritmos basados en la teoría de la evolución de Darwin son capaces de encontrar soluciones potencialmente óptimas sin necesidad de explorar todos los estados posibles, como es el caso de los planificadores basados en búsqueda. Este tipo de algoritmo se encarga de representar las soluciones como individuos de una población según el nivel de adaptación que estos alcancen, respecto de las tareas que han de cumplir.

Por ello este proyecto se ha dedicado a desarrollar un sistema de planificación de misiones basado en algoritmos genéticos.

Capítulo 3

Modelo de escenario de misiones

En este capítulo se muestra toda la información sobre la estructura desarrollada para modelar todos los elementos que aparecen en las misiones, es decir, UAVs, zonas, sensores y tareas.

En la Figura 3.1 se muestra el esquema del modelo de datos utilizado para la carga y generación de misiones que crearán el problema a resolver. Tanto la colección de UAVs, sensores que estos pueden incorporar, como las zonas que pueden visitarse en las misiones, se obtienen a partir de una base de datos que almacena toda la información. Por otro lado, las tareas a realizar son insertadas por el usuario a un fichero XML donde se especifican las tareas y las zonas donde se debe realizar estas.

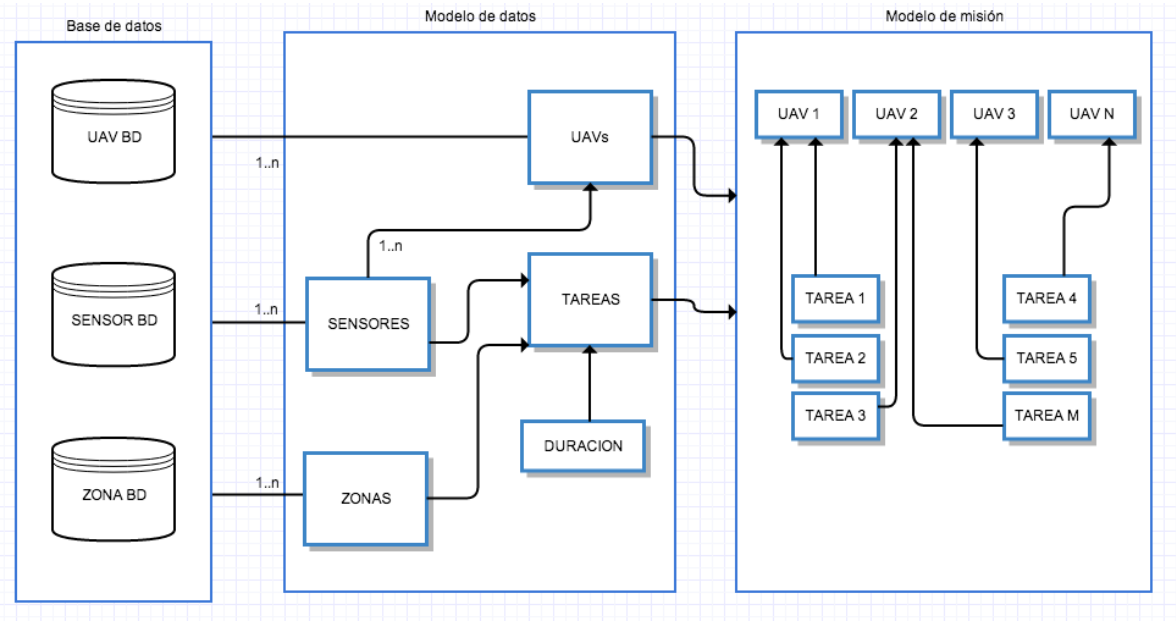

Figura 3.1: Modelo de datos de la aplicación

3.1. Descripción de la misión

El problema que se plantea resolver será dar una solución óptima a la misión planteada. Cada misión viene determinada por una serie de tareas proporcionadas por el usuario que se deben realizar. En la solución final, todas y cada una de las tareas deberán ser asignadas a los UAVs de la colección, de tal manera que estos puedan realizar cada tarea asignada de forma exitosa.

Una misión solo se puede completar si todas y cada una de las tareas que la componen han sido asignadas a algún UAV disponible y, dicho UAV, tiene la capacidad o el equipamiento necesario para llevarla a cabo. Para ello se realiza una evaluación de las tareas en dos pasos por medio de la función fitness, como se explica en la sección 4.2.

La resolución de todas las misiones sigue el diagrama de la Figura 3.2. Como se puede observar como parámetros de entrada se reciben una serie de tareas asignadas a una zona en concreto. Por otro lado, la colección de UAVs utilizada será cargada desde la base de datos presente en el sistema.

Tras la ejecución del algoritmo genético para una misión recibida se tiene una salida con tuplas de tres elementos con la forma $[T_i, Z_j, UAV_k]$, siendo T_i una tarea, Z_j una zona donde se debe realizar la tarea, y UAV_k un UAV de la colección con capacidad de realizar la tarea T_i en la zona Z_j cumpliendo todos los criterios.

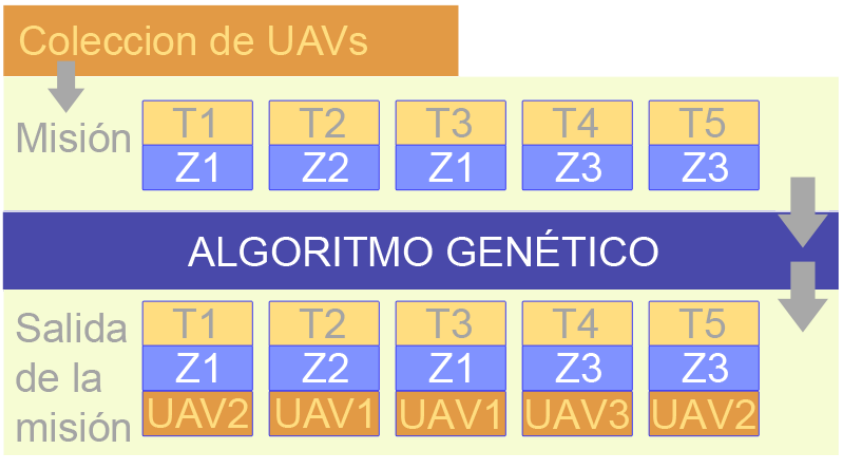

Figura 3.2: Diagrama de flujo de una misión

A continuación se muestra un listado de tablas donde se representa un posible escenario de misión a resolver representada en la tabla 3.1, junto con la lista de UAVs disponibles para realizar dicha misión mostrada en la tabla 3.2. La tabla 3.3 muestra una de las soluciones posibles para los datos recibidos.

En este caso la solución ha optado por asignar la tarea 1 al UAV4 en lugar del UAV1, esto se debe a que este último es el único capaz de realizar la tarea 4, por ello se le ha asignado a

ID de Tarea	Sensor	Duración	Zona	Descripción
T1	Electrooptico	20min	Zona A	Capturar imágenes de alta definición de la zona.
T2	Infrarrojos	20min	Zona A	Tomar fotografias térmicas de la zona A
T3	Carga	11:00 a 11:45	Zona B	Realizar tarea de recogida de carga en la Zona B.
T4	Electrooptico	9:00 a 15:00	Zona C (Restringida)	Videovigilancia de la zona C (Restringida)

Tabla 3.1: Ejemplo de entrada de tareas

ID de UAV	Sensores	Acceso restringido
UAV1	$Electrooptico \vert Carga$	SI
UAV2	$Infrarrojos \vert Electrooptico$	SI
UAV3	$Carga \vert SAR$	NO
UAV4	$Electrooptico \vert SAR \vert Carga$	NO

Tabla 3.2: Colección de UAVs para realizar una misión

ID de tarea	ID de UAV
T1	UAV4
T2	UAV2
T3	UAV3
T4	UAV1

Tabla 3.3: Posible resolución de la misión

este UAV debido a su permiso de entrada a zonas restringidas.

Por otro lado la tarea 2 ha sido asignada al UAV2 ya que es el único UAV con el equipo necesario para realizar la tarea en cuestión.

Y finalmente, la tarea 3 podría ser realizada indistintamente por el UAV1 y el UAV3, pero se ha seleccionado el UAV3 ya que permite completar la misión general en menos tiempo al realizar más tareas en paralelo de forma coordinada.

3.1.1. Datos de misión

En esta sección se detalla la estructura propuesta para la construcción de las misiones que recibirá el algoritmo genético.

Cada misión tiene asignadas un número N de tareas, además posee toda la colección de datos que tiene almacenados el sistema mediante la BBDD, tanto UAVs, Sensores, Zonas. (Detallado en la sección 3.2).

Como se ha explicado previamente, para que una misión sea satisfactoria se deben cumplir todas las restricciones de las tareas recibidas, además de cumplir las restricciones de entrada de las zonas asignadas a dicha tarea. Estas restricciones serán detalladas de forma individual en la sección 4.2.

El sistema esta desarrollado para poder realizar tareas de forma simultánea con los diferentes UAVs de la colección. Como se puede ver en la Figura 3.3 se muestra una línea de tiempo general de la misión donde aparecen las diferentes tareas asignadas a cada UAV y a su vez el tiempo que tarda cada UAV en realizar sus tareas.

De esta manera, el tiempo de la misión queda determinado por un intervalo de tiempo donde el primer UAV comienza sus tareas hasta que todos y cada uno de los UAV han finalizado todas sus tareas de forma satisfactoria.

En esta misma figura podemos ver como en un margen de 24 horas se han realizado 14 tareas de forma simultánea, por una colección de 4 UAVs diferentes.

3.2. Modelo de datos

Esta sección se centra en la descripción detallada de cada uno de los elementos que forman el sistema de resolución de misiones, en este punto se explicarán qué atributos posee cada uno de estos elementos y qué tipo de valores puede contener cada uno de estos.

3.2.1. Modelo UAV

En esta sección se muestra la información seleccionada para representar los UAV en el sistema. En la siguiente lista se muestran los atributos más relevantes en la ejecución de la misión.

- **Consumo de combustible**: Este atributo determina el gasto de combustible por unidad de distancia de cada UAV.

Figura 3.3: Línea de ejecución de las tareas de la misión

- **Estado**: Esta estructura es utilizada para controlar el estado actual de los UAV, y comprobar si son UAVs válidos para realizar las tareas. Estos estados son los siguientes:
  - Libre: El UAV esta disponible para asignarle tareas.
  - Dañado: El UAV tiene algún problema en su funcionamiento y no garantiza poder realizar la tarea de forma correcta.
  - Ocupado: El UAV esta ocupado en este momento y no puede recibir nuevas tareas hasta que complete las actuales.
  - Destruido: El UAV ha sufrido daños irreparables y no puede realizar ninguna tarea.
- **Posición**: Este objeto es utilizado para determinar su localización geográfica mediante coordenadas geográficas (latitud, longitud y altura). Mediante este atributo se puede calcular la distancia entre dos puntos mediante la siguiente fórmula[1].

$$\alpha = \sin\left(\frac{(Lat1 - Lat2)}{2}\right)^2 + \sin\left(\frac{(Lon1 - Lon2)}{2}\right)^2 \cdot \cos(Lat1) \cdot \cos(Lat2) \tag{3.1}$$

$$\beta = 2 \cdot \arctan(\sqrt{\alpha}, \sqrt{1-\alpha}) \tag{3.2}$$

$$Dist_{Km} = R \cdot \beta \tag{3.3}$$

[1]Siendo R el radio terrestre con valor 6371Km

- **Lista de tareas**: Este atributo almacena la colección de tareas que un UAV tiene asignadas durante la misión. Todas las tareas quedan insertadas por orden de horario.

Por otro lado cada vehículo posee una colección de métodos que devuelven información sobre la misión que tienen asignada o sobre el estado o valor actual de sus atributos, estos son los siguientes:

- **Duración de misión**: Devuelve la duración total de la misión formada por subtareas en el momento actual del UAV. Este intervalo de tiempo comienza con el inicio de la primera tarea y termina con la fecha de finalización de la última tarea con fecha asignada, además del tiempo de las tareas que no tienen tiempo asignado.
- **Añadir tareas**: Este método asigna nuevas tareas a un UAV según el orden cronológico de estas (tras la asignación se produce una reordenación de las tareas).
- **Distancia de la misión**: Este método calcula la distancia que debe recorrer el UAV para alcanzar todas las tareas que tiene asignadas. Como punto de partida establece su posición actual y se calcula la distancia entre tareas según su orden de ejecución.

Representacion de UAVs en base de datos

Los UAVs almacenados en el sistema queda definidos de la siguiente manera (en la representación quedan resaltados en azul oscuro los atributos propios de objeto **UAV**):

```
<UAV>
    <autonomy>2</autonomy>
    <fuelConsumption>4.0</fuelConsumption>
    <state>IDLE</state>
    <maxAltitude>8</maxAltitude>
    <maxSpeed>40</maxSpeed>
    <position>
      <longitude>10.05</longitude>
      <latitude>0.0</latitude>
      <altitude>10.0</altitude>
    </position>
    <initFirstTask>292278994-08-17 07:12:55.807 UTC</initFirstTask>
    <finishLastTask>292269055-12-02 BC 16:47:04.192 UTC</finishLastTask>
    <nombre>0</nombre>
    <sensorList>
      <Sensor>
        <sensorType>SAR</sensorType>
        <weightCapacity>0</weightCapacity>
        <zoom>30</zoom>
        <scope>50</scope>
      </Sensor>
      <Sensor>
        <sensorType>ELECTROOPTICAL</sensorType>
        <weightCapacity>0</weightCapacity>
        <zoom>100</zoom>
        <scope>100</scope>
      </Sensor>
    </sensorList>
```

```
29      <restriction>true</restriction>
    </UAV>
```

3.2.2. Modelo Sensor

Todos los UAVs van equipados con uno o más sensores con los que realizan las tareas asignadas. Estos sensores puede ser de distintos tipos y, a su vez, dentro de cada tipo de sensor poseen diferentes atributos que determinan algunos aspectos como su alcance, su potencia o su capacidad. A continuación se muestra un listado de los sensores con los que están equipados los UAV del sistema propuesto:

- **Sensor electroóptico**: Este tipo de sensor proporciona la capacidad de realizar fotografías de gran amplitud y larga distancia a los UAV que disponen de él. En la figura 3.2.2 se muestra un micro UAV equipado con una cámara basada en sensor electroóptico.

- **Sensor infrarrojo**: Este tipo de sensor proporciona al UAV que lo equipa la capacidad de realizar fotografías y vídeos por la noche o en condiciones de muy baja luminosidad. Estos sensores también son capaces de realizar fotografías térmicas, especialmente utilizadas en el análisis zonas forestales para la predicción y prevención de incendios. En la figura 3.2.2 aparece un ejemplo de fotografía térmica tomada por un sensor infrarrojos.

- **Sensor SAR**: El sensor SAR (*Synthetic Aperture Radar*) es un sensor capaz de capturar imágenes mediante tecnología de antenas radar. Este tipo de radar consigue mediante varios barridos simular un barrido virtual de un radar de gran tamaño. Sin embargo, las antenas necesarias para ello son mucho menores, por lo que, son perfectas para ser incorporadas en UAV pequeños de reconocimiento. En la figura 3.2.2 aparece una imagen tomada mediante un sensor SAR de la superficie de Venus.

- **Sensor de carga**: Este tipo de sensor establece la cantidad de carga que puede transportar el UAV. Dependiendo del vehículo la carga puede variar en gran medida, desde menos de 100 gramos de peso hasta cargar con varias toneladas de peso.

Representacion de Sensores en base de datos

Los sensores quedan representados en la base de datos de la siguiente forma (en la representación quedan resaltados en azul oscuro los atributos propios de objeto **Sensor**):

```
<Sensor>
2     <sensorType>ELECTROOPTICAL</sensorType>
      <weightCapacity>0</weightCapacity>
4     <zoom>100</zoom>
      <scope>100</scope>
6   </Sensor>
```

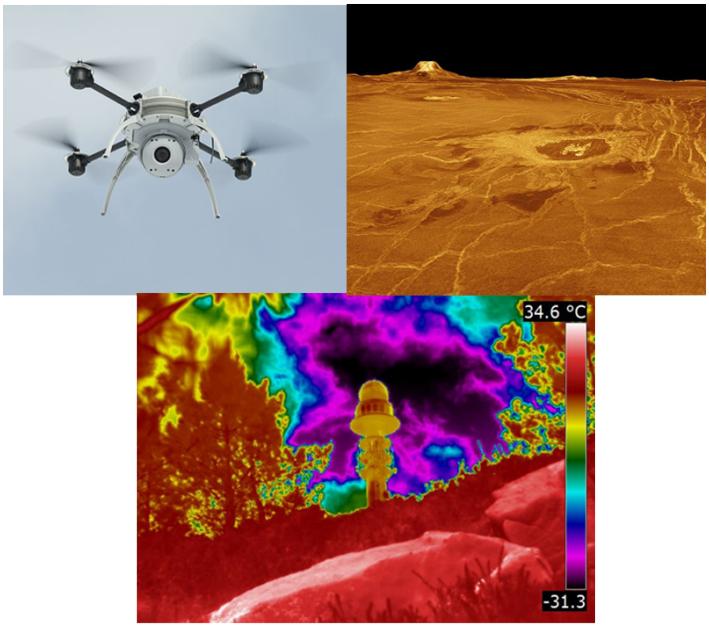

Figura 3.4: Sensores UAV (Electroóptico, SAR, Infrarrojo)

3.2.3. Modelo Zona

Todas las tareas presentes en cada misión deben realizarse en una zona determinada. El sistema trabaja con zonas extraídas de una base de datos donde previamente se han incorporado los datos de estas.

Cada zona tiene una serie de atributos base que las define, estos son:

- **Nombre**: El nombre que define como única a una zona, este atributo funciona como identificador de ella, por lo que no deben existir dos nombres de zona iguales.
- **Descripción**: Este atributo contiene una breve descripción de la zona, además de información relevante que se deba tener en cuenta a la hora de asignar tareas en esta localización.
- **Posición**: Este atributo determina el centro de la zona. Como ocurre con los UAV este atributo está definido en forma de coordenadas geográficas (latitud, longitud y altura). Sigue la misma fórmula para el cálculo de distancias que la definida en el atributo posición de los UAV en el apartado 3.2.1.
- **Radio**: Determina el radio de la zona desde el punto central del atributo posición.
- **Restricción de entrada**: Este atributo determina si la zona está marcada como restringida, por lo que solo los UAV con permisos de entrada podrán sobrevolarla y realizar

misiones en el área que esta zona comprende. En la figura 2.3 se aprecia el comportamiento de un UAV sin permiso de entrada a zonas restringidas.

Representacion de Zonas en base de datos

El modelo que representa las zonas en la base de datos tiene la siguiente forma (en la representación queda resaltado en azul oscuro los atributos propios del objeto **Zona**):

```
<Zone>
    <zoneName>Zona 0</zoneName>
    <description>La zona A tiene forma circular, y no tiene restriccion de
        entrada</description>
    <restrictedZone>false</restrictedZone>
    <position>
      <longitude>10.01</longitude>
      <latitude>0.0</latitude>
      <altitude>10.0</altitude>
    </position>
    <radius>30</radius>
  </Zone>
```

3.2.4. Modelo Tarea

Las tareas son el elemento principal que componen las misiones, dado que cada misión se compone un número de tareas que los UAV deben completar. Todas las tareas tienen la misma estructura que consta de los siguientes atributos:

- **Zona**: Cada tarea está asignada a una zona almacenada del sistema. Para que la tarea pueda ser llevada a cabo en la zona especificada, el UAV deberá tener permisos de entrada si esta zona esta marcada como zona **restringida** (esta restricción está definida en la sección 3.2.3).
- **Sensor**: Este atributo determina qué sensor ha de utilizarse una vez el UAV alcance la posición determinada por el atributo de **zona**. En este caso, para que la tarea pueda ser completada, el UAV debe poseer el sensor y que este tenga la capacidad de realizarla.
- **Duración**: Este atributo determina la duración total de la tarea. Esta duración puede venir determinada de tres maneras:
  - 1. Fecha de inicio y fecha de fin de tarea
  - 2. Fecha de inicio y duración de la tarea
  - 3. Duración total de la tarea[2]

  Las tareas tienen preferencia según el tipo de duración que haya sido establecida, siendo más importantes las tareas que tienen una fecha concreta para su realización. Mientras que las tareas que tienen establecida únicamente su duración, se resolverán cuando se hayan finalizado todas las tareas de prioridad más alta según el orden de asignación.

[2]La duración de las tareas puede ser establecida en segundos, minutos, horas o días indistintamente

La duración de las tareas se gestiona mediante un atributo llamado *datePair* con el que cada tarea queda determinada con una fecha de inicio y una fecha de fin. Mediante este atributo se puede comprobar si dos tareas son compatibles para ser realizadas por un mismo UAV.

3.3. Modelo de base de datos

El sistema de resolución de misiones obtiene datos a partir de una base de datos donde se almacenan todos los UAVs disponibles para las misiones, las zonas disponibles para realizar las tareas y los sensores con los que los UAVs pueden ser equipados.

Esta base de datos esta construida mediante ficheros de etiquetado XML, estos objetos son serializados a través de la librería de código abierto con licencia BDS **XStream** de Java[3].

[3]http://xstream.codehaus.org/

Capítulo 4

AG para la planificación de misiones

En este capítulo se expone toda la información referente a la implementación de un algoritmo genético para la resolución de misiones con UAVs en paralelo.

El algoritmo desarrollado sigue la estructura de un algoritmo genético simple siguiendo la estructura propuesta en el pseudocódigo 1 de la sección 2.1.1. Para el desarrollo del algoritmo se ha utilizado la biblioteca de clases **ECJ** (Evolutionary Computation Java) desarrollado por el *ECLab* de la universidad George Mason[1].

4.1. Codificación del AG

En esta sección se desglosa la codificación de los cromosomas que se ha seleccionado para representar el problema de las misiones.

Para la codificación de los cromosomas se ha optado por un vector de números enteros donde cada valor representa cada uno los UAV de la base de datos. Por otro lado, cada posición del vector representa la tarea que se debe realizar. De esta manera, si el problema plantea una misión donde se deben resolver N tareas, con una colección de M UAVs, tendremos un vector de N posiciones con un identificador en su interior que puede contener valores de 0 a M-1, como se muestra en la figura 4.1.

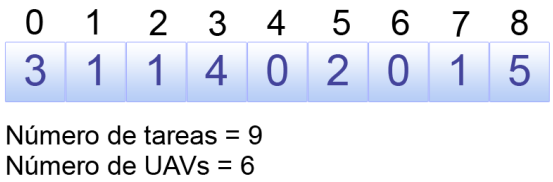

Figura 4.1: Ejemplo de un cromosoma en el algoritmo genético

[1] Web del proyecto ECJ: `http://cs.gmu.edu/~eclab/projects/ecj/`

En este caso la solución de la figura 4.1 se traduce de la siguiente manera:

- **UAV0**: Al UAV 0 le ha sido asignado el conjunto de tareas $[4, 6]$
- **UAV1**: Al UAV 1 le ha sido asignado el conjunto de tareas $[1, 2, 7]$
- **UAV2**: Al UAV 2 le ha sido asignado el conjunto de tareas $[5]$
- **UAV3**: Al UAV 3 le ha sido asignado el conjunto de tareas $[0]$
- **UAV4**: Al UAV 4 le ha sido asignado el conjunto de tareas $[3]$
- **UAV5**: Al UAV 5 le ha sido asignado el conjunto de tareas $[8]$

En este ejemplo no se ha especificado ningún tipo de restricciones temporales a las tareas, esto significa que los UAV que posean más de una tarea (en este caso el UAV0 y el UAV1) las realizarán de forma secuencial y consecutiva. Mientras que las tareas 4, 1, 5, 0, 3 y 8 se realizarán de forma paralela al estar asignadas cada una a un UAV diferente.

Se deben cumplir todos los criterios de validez descritos en la sección 4.2.4 para poder realizar el proceso de asignación de tareas. Una vez superados estos criterios, las tareas se insertarán ordenadas según su fecha de inicio y, en caso de no tener fecha establecida, se encolarán detrás de la última tarea asignada.

4.2. Función de ajuste (Fitness)

4.2.1. Descripción

Una función de ajuste o *fitness* es un procedimiento mediante el cual un elemento es evaluado según un conjunto de uno o más criterios. Según la eficacia con la que el elemento cumpla los citados criterios, recibirá una puntuación.

En este caso la función fitness desarrollada en este proyecto consta de dos fases diferenciadas para evaluar individuos capaces de resolver el problema que se esté tratando.

Evaluación de criterios de validez o validación de solución: En este punto, se comprueba que cada una de las tareas es compatible con el UAV al que ha sido asignado asignado y que es posible realizarlas en los plazos de tiempos que vienen determinados. Si estas condiciones no se cumplen, el individuo queda automáticamente descartado, recibiendo la puntuación de evaluación más baja, en este caso 0.

Después de ser evaluado por este tipo de criterios, todos los individuos que han sido capaces de terminar con éxito este primer paso, pasarán a realizar la evaluación de criterios de calidad.

Evaluación de criterios de calidad: En segundo lugar, se han de evaluar los requisitos de la tarea (lugar de destino, sensor a utilizar, restricción de zona, etc.) acorde a las características del UAV, de tal manera que la misión reciba una puntuación final con la que poder realizar una clasificación de los individuos que posteriormente será tratada por el operador de selección.

Los criterios de **calidad** de la solución planteados en este proyecto han sido diseñados para que las puntuaciones estén acotadas en el intervalo $[0, 1]$, donde 0 es considerado como la peor

puntuación posible y 1 la puntuación óptima para un individuo. Algunos de los criterios de calidad calculados se basan en atributos de los UAV, como el consumo de combustible en la misión, el tiempo que durará el total de la misión o la distancia que los UAV deben recorrer para alcanzar todas las tareas.

4.2.2. Fitness utilizada (SimpleFitness)

En este proyecto se ha optado por utilizar la función *SimpleFitness* ofrecida por ECJ. La cual se fundamenta en crear una clasificación de individuos basados en una puntuación representada por un valor decimal **maximizado**, es decir, donde el valor más alto es el mejor posicionado en el ranking de individuos.

Por defecto esta función no establece un límite en el valor decimal de cada individuo, pero para poder evaluar atributos heterogéneos en conjunto (distancia, tiempo, combustible, etc), se ha optado por realizar una normalización de las evaluaciones de estos.

Todas las normalizaciones están realizadas para obtener valores dentro del intervalo $[0, 1]$. Para ello, en todas las funciones de evaluación se ha realizado una estimación del caso mejor posible, de tal manera que el resultado de esta función mantenga el concepto de la siguiente fórmula:

$$\alpha = \frac{BestFitness}{FitnessValue} \text{ donde } \alpha \in [0, 1] \tag{4.1}$$

Esta función recibe un parámetro con el que se especifica qué valor posee un individuo considerado óptimo. El propósito de este parámetro es finalizar la evaluación en caso de encontrar el individuo perfecto antes de finalizar la condición de convergencia o el número establecido de generaciones para el fin de la ejecución.

Algoritmo 2: Pseucodcódigo del cálculo de función fitness

Input: $M = (X_1, X_2..., X_n)$ donde $X_i = [U_x, [T_i, Z_y]]$ siempre que $U \in$ UAV, $T \in$ Tarea y $Z \in$ Zona

Output: f_{val} Valor decimal en el intervalo $[0, 1]$ para el conjunto I

$M \leftarrow$ Conjunto de tuplas de UAVs con tareas asignadas que forman la misión

$f_{val} \in [0, 1]$ Valor de la fitness global de la solución para la misión; Ci Cada uno de los criterio de validez

Wi Peso asociado a cada uno de los criterios de calidad ci

for $i \leftarrow 0$ ***to*** $M_{size} - 1$ **do**

  $Ind_{OK} = \prod_{i=0}^{n} C_i(M)$

if Ind_{OK} **then**

  $f_{val} = \sum_{i=0}^{n} c_i(M) \cdot W_i$

else

  return *0*

return f_{val}

4.2.3. Cálculo de la fitness

La fitness del algoritmo se calcula mediante la unión de todos los criterios de validez descritos anteriormente, de tal manera que un individuo que cumpla todos los criterios de validez tendrá una puntuación de 0 a 1, mientras que los individuos que no cumplan alguno de estos criterios harán la solución nula y en consecuencia recibirán una puntuación de 0.

La fitness desarrollada para este proyecto está representada por las siguientes fórmulas para cada misión M generada por el algoritmo.

Siendo cualquier X_i tuplas del tipo $[U_x, [T_i, Z_y]]$, con $\alpha = 0$ o $\alpha = 1$

$$\alpha = \prod_{i=0}^{n} containsSensor(X_i) \cdot evaluateTaskByHour(X_i) \cdot isRestrictedArea(X_i) \tag{4.2}$$

Y ahora

$$F_M = (Distance \cdot w_{dist} + Duration \cdot w_{dur} + Fuel \cdot w_{fuel} + State \cdot w_{state}) \cdot \alpha \tag{4.3}$$

Para el cálculo final de F_M, cada uno de los criterios de calidad de la solución que se evalúan tiene un coeficiente de peso w asociado a ellas. Este peso determina la importancia que tiene cada criterio de calidad en el valor total de la fitness.

Para este proyecto, los valores de estos coeficientes se han calculado mediante experimentación, seleccionando los que mejores resultados de misiones han ofrecido tras múltiples ejecuciones.

A continuación, se detalla el funcionamiento de cada uno de los criterios utilizados para el cálculo global de la función de fitness.

4.2.4. Criterios de la función fitness

Criterios de validez

Este tipo de criterios garantiza que la misión pueda ser resuelta con éxito, descartando cualquier asignación que imposibilite la realización de la misión.

Sensor de la tarea: Esta evaluación se encarga de comprobar que el UAV al que se le asigna la tarea posee el sensor o equipamiento necesario para realizar dicha tarea.

Zona restringida: Este tipo de criterio comprueba si la zona donde debe realizarse la tarea tiene restricción de entrada y, en caso afirmativo, comprueba que el UAV que debe realizar la tarea tiene permiso de entrada a zonas restringidas.

Horario de tareas: Este tipo de evaluación restringe las tareas que son asignadas a los UAVs según el horario de ejecución que estas posean. Por ejemplo, dada una tarea *T1*[10:00AM-11:00AM] y otra*T2*[10:30AM-11:30AM], existe una restricción temporal en el caso de que sean asignadas al mismo UAV, ya que no pueden realizarse en las horas establecidas.

Algoritmo 3: Evaluación de horario de Tareas

Input: T siendo la tarea a evaluar, U siendo el UAV donde se va a asignar la tarea
Output: 1 en caso de poder asignar la tarea, 0 en caso contrario

```
for i ← 0 to U_TaskSize do
    if T_date > U_tinit AND T_date < U_tfinish then
        return 0;
return 1
```

Criterios de calidad de la solución

Finalmente, si un individuo es válido se tendrá que valorar el grado de eficacia con la que los UAVs son capaces de realizar las tareas que tienen asignadas en la misión. Se van a tener en cuenta tres criterios de calidad diferentes para medir cómo de buena es una solución[2].

Distancia de la misión: Evalúa la distancia que deben recorrer los UAVs respecto de la distancia estimada como ideal para toda la misión. Se va a considerar que cuanta menos distancia total se tenga que recorrer, la solución para la misión será mejor considerada.

Algoritmo 4: Evaluación de distancia

Output: Valor entre 0 y 1 que representa la cercanía respecto de la mejor distancia estimada

```
Dist = 0
for i ← 0 to U_collection do
    Dist += getDistance(U_i, U_t0)
    for i ← 1 to U_taskList do
        Dist+ = getDistance(U_ti−1, U_ti)
return BestDistance/Dist
```

Duración total de la misión: Se evalúa el tiempo total de duración de la misión, respecto de una estimación del tiempo ideal para realizar dicha misión. Para realizar la estimación del mejor tiempo de misión, se tienen en cuenta todas las duraciones de las tareas recibidas y se selecciona la que posea la mayor duración de todas de forma individual, esto se debe a que ninguna misión puede resolverse en un tiempo menor que la tarea más larga por la que esté compuesta.

Consumo de combustible: Este criterio calcula la eficiencia de los UAVs al realizar sus tareas según el consumo de combustible que necesitan para realizarlas. Para esto, se compara con una estimación ideal del consumo de combustible. Dicha estimación toma como mejor opción el consumo del UAV que menos combustible gasta por unidad de distancia.

[2]Dado que los atributos evaluados en los criterios de calidad son muy heterogéneos entre ellos, todos han sido normalizados para que la evaluación quede acotada en el intervalo $[0, 1]$, según la fórmula 4.2

Algoritmo 5: Evaluación de duración

Output: Valor entre 0 y 1, según la diferencia con la mejor duración estimada

```
1  initDate = V_MAX
2  finishDate = V_MIN
3  duration = 0
4  for i ← 0 to U_collection do
5      for i ← 0 to U_taskList do
6          if getInitDate(Uti) = ∅ then
7              duration += getDuration(Uti)
8          else
9              initDate = min(Uti, initDate)
10             finishDate = max(Uti, finishDate)
11 Duration = (finishDate - initDate) + duration
12 return bestDuration/Duration
```

4.3. Algoritmo desarrollado

Como se ha mencionado al comienzo de este capítulo, para desarrollar este sistema se ha utilizado una librería de clases Java especializada en el modelado de algoritmos genéticos, **ECJ**. Esta librería ofrece una gran cantidad de opciones diferentes para crear múltiples tipos de algoritmos genéticos dependiendo del nivel de complejidad que se desee.

Para el caso de la planificación de misiones, se ha optado por modelar un **algoritmo genético simple**, debido a que nuestro modelo puede ser representado mediante un cromosoma basado en números enteros como se explica en la sección 4.1. Con esto conseguimos que el modelado del problema no incremente la complejidad del algoritmo y sea únicamente la función de fitness donde se focalice toda la complejidad del algoritmo.

4.3.1. Selección

En esta sección se explica el funcionamiento del operador de selección *TournamentSelection* implementado por ECJ en este algoritmo genético de tipo Simple. El funcionamiento de la selección por torneo es el siguiente:

Partiendo de una población de P (population.size = P) individuos, se debe establecer un tamaño de torneo al que denominaremos N (select.tournament.n = N). De esta manera, se genera un torneo de tamaño N, formado por los mejores individuos de la población.

En este punto se selecciona el tamaño del enfrentamiento S (select.tournament.size = S). Con este parámetro se realizará un enfrentamiento entre S individuos escogidos entre los participantes del torneo, donde el individuo con el valor de fitness mayor saldrá vencedor. En la Figura 4.2 se muestra un ejemplo gráfico de este operador.

Los parámetros de ECJ aplicados en el algoritmo para configurar este operador son los siguientes:

- **Tamaño del torneo**: Se seleccionará el 20 % de los mejores individuos de la población P

Figura 4.2: Ejemplo selección por torneo

(select.tournament.n = 0.20).

- **Individuos enfrentados**: Se enfrenta a 4 de los individuos participantes en el torneo y así se obtiene el individuo con la fitness de mayor valor. En caso de empate se seleccionará un individuo de manera aleatoria (select.tournament.size = 4).

Además en el sistema se ha utilizado la versión elitista de la selección que, como se explica en la sección 2.1.1, se basa en seleccionar automáticamente N individuos sin ser cruzados con otros.

Esto se realiza mediante el parámetro **breed.elite-fraction.0 = 0.10** que, en este caso, promociona al 10 % de los individuos mejor adaptados de la población.

4.3.2. Cruce

En esta sección se detalla el tipo de operador de cruce utilizado para el algoritmo genético planteado. En este caso, el cruce será del tipo *One point*. Este tipo de cruce del algoritmo genético simple se basa en escoger dos padres a los que les selecciona el mismo punto del genoma. Entonces, intercambia la información de ambos padres para generar dos nuevos descendientes con el genotipo recombinado.

A diferencia del ejemplo de cruce en dos puntos que muestra en la Figura 2.1, este únicamente divide el genoma del individuo en dos partes, como se muestra en la Figura 4.3.

Los parámetros de ECJ aplicados para configurar este operador son los siguientes:

- **Tipo de cruce**: Se ha seleccionado este tipo de cruce mediante experimentación, ya que, siendo el más sencillo es el que mejores resultados ha ofrecido en ejecución (pop.subpop.0.species.crossover-type = one).

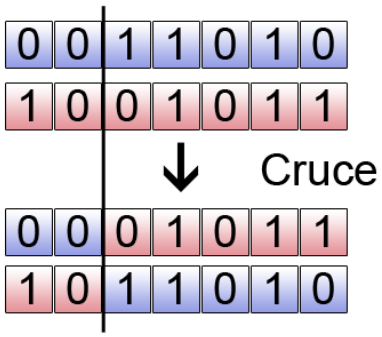

Figura 4.3: Ejemplo cruce *One point*

- **Probabilidad de cruce**: En este caso la probabilidad es 1 debido a que para promover la diversidad de individuos se obliga al cruce en cada generación (pop.subpop.0.species.crossover-prob = 1.0).

4.3.3. Mutación

En esta sección se desarrolla el tipo de mutación utilizada en el algoritmo genético a partir de los tipos que ofrece ECJ.

Para este tipo de algoritmo, se ha utilizado la mutación de tipo *uniform*. El funcionamiento de esta mutación se basa en realizar un cambio en uno de los genes del individuo cambiando su valor de manera aleatoria. Aplicado a este problema, el valor al que puede ser cambiado está acotado en el intervalo del número de UAVs que haya en el sistema.

La mutación es un parámetro principalmente utilizado para garantizar más diversidad de individuos y evitar la convergencia prematura. Por otro lado la mutación puede causar efectos nocivos si ocurre con demasiada frecuencia, por ello ECJ proporciona un parámetro para especificar la probabilidad con la que puede ocurrir dicha mutación.

Los parámetros de ECJ aplicados en este algoritmo para configurar este operador son los siguientes:

- **Tipo de mutación**: Se ha seleccionado este tipo de mutación mediante experimentación debido a que los resultados obtenidos en la fitness han sido más altos frente a los otros tipos de mutación (pop.subpop.0.species.segment.0.mutation-type = reset).
- **Probabilidad de mutación**: En este caso la probabilidad es 0.1 ya que, como se ha comentado, la mutación no debe ocurrir muy frecuentemente (pop.subpop.0.species.mutation-prob = 0.1).

De esta manera, en el sistema planteado donde se ha establecido una poblacion de 1000 indviduos, cada nueva generación se forma mediante:

- 100 individuos promocionados de forma directa.

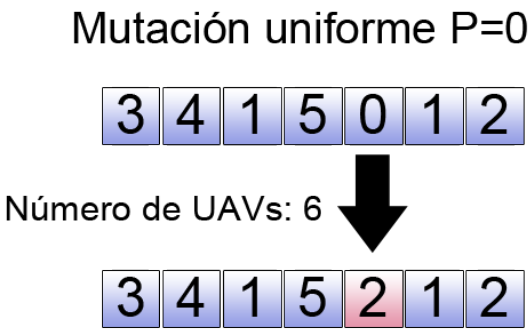

Figura 4.4: Ejemplo de mutación *Uniform*

- 900 individuos generados a partir de aplicar los operadores de selección, cruce y mutación.

Capítulo 5

Experimentación y resultados obtenidos

5.1. Descripción del conjunto de datos

5.1.1. Tareas

En esta sección se muestra la colección total de tareas que existe en el sistema y con las que se han realizado todas las pruebas. Además, estas tareas se utilizarán en secciones posteriores para mostrar la evaluación de resultados.

ID de Tarea	Sensor	Duración	Zona	Descripción
T0	Infrarrojos	00:00 a 00:10	Zona B	Utilizar el sensor infrarrojo.
T1	SAR	00:10 a 00:20	Zona C	Utilizar el sensor SAR.
T2	Carga	00:20 a 00:30	Zona D	Utilizar el sensor Carga.
T3	Electroóptico	00:40 a 00:50	Zona A	Utilizar el sensor Electroóptico.
T4	Electroóptico	15 min	Zona F	Utilizar el sensor Electroóptico.
T5	SAR	25 min	Zona G	Utilizar el sensor SAR.
T6	Carga	00:10 a 00:20	Zona I	Utilizar el sensor Carga.
T7	Infrarrojos	15 min	Zona I	Utilizar el sensor infrarrojo.
T8	Infrarrojos	32 min	Zona J	Utilizar el sensor infrarrojo.
T9	Infrarrojos	15 min	Zona C	Utilizar el sensor infrarrojo.
T10	SAR	00:50 + 15 min	Zona K	Utilizar el sensor SAR.
T11	Infrarrojos	35 min	Zona A	Utilizar el sensor infrarrojo.
T12	Electroóptico	1:00 + 23 min	Zona L	Utilizar el sensor Electroóptico.
T13	Electroóptico	00:40 a 1:00	Zona A	Utilizar el sensor Electroóptico.
T14	Carga	50 min	Zona I	Utilizar el sensor Carga.

Tabla 5.1: Colección de tareas del sistema

5.1.2. Vehículos

En esta sección se muestra toda la colección de UAVs que existe en el sistema. Con estos se resolverán las misiones planteadas. Estos UAVs son el conjunto del que posteriormente se mostrarán datos de evaluación de resultados.

Nombre	Estado	Sensores	Posición	Consumo	Ac.Restr
UAV0	Disponible	Electroóptico|SAR	(0, 10.05, 10)	$4l/km$	SI
UAV1	Disponible	Electroóptico|$Infrarrojo$|$Carga$	(0, 10.05, 10)	$8.5l/km$	NO
UAV2	Disponible	Electroóptico|$Infrarrojo$|SAR	(0, 10.05, 10)	$3.1l/km$	NO
UAV3	Disponible	Electroóptico|$Infrarrojo$|SAR|$Carga$	(0, 10.05, 10)	$9l/km$	NO
UAV4	Disponible	Electroóptico|$Carga$	(0, 10.05, 10)	$2l/km$	SI
UAV5	Disponible	Infrarrojo|SAR|$Carga$	(0, 10.05, 10)	$2.5l/km$	SI

Tabla 5.2: Colección de UAVs del sistema

5.1.3. Zonas

En esta sección se muestran todas las zonas disponibles para realizar las tareas. Con estas zonas se han realizado las pruebas de ejecución y serán utilizadas en los resultados de las siguiente secciones.

Zona	Restricción	Posición	Radio	Descripción
Zona A	NO	(0, 10.01, 10)	30	Descripción Zona A
Zona B	NO	(0, 10.02, 10)	50	Descripción Zona B
Zona C	SI	(0, 10.03, 10)	10	Descripción Zona C
Zona D	SI	(0, 10.04, 10)	10	Descripción Zona D
Zona E	NO	(0, 10.05, 10)	15	Descripción Zona E
Zona F	NO	(0, 10.06, 10)	50	Descripción Zona F
Zona G	NO	(0, 10.07, 10)	20	Descripción Zona G
Zona H	NO	(0, 10.08, 10)	32	Descripción Zona H
Zona I	NO	(0, 10.09, 10)	21	Descripción Zona I
Zona J	SI	(0, 10.1, 10)	54	Descripción Zona J
Zona K	NO	(0, 10.11, 10)	34	Descripción Zona K
Zona L	NO	(0, 10.12, 10)	100	Descripción Zona L

Tabla 5.3: Colección de zonas del sistema

5.2. Análisis preliminar para determinar pesos de la función fitness

Como se puede observar en la fórmula del cálculo de la función fitness explicada en el apartado 4.2, cada uno de los criterios de calidad que la conforman posee un coeficiente de peso utilizado para regular la relevancia en el valor final de la fitness.

Dado que estos coeficientes determinan en gran medida la eficacia total de la fitness, es fundamental encontrar los valores que consiguen incrementar su eficacia a los valores más altos, pues esto se traduce en conseguir soluciones óptimas al problema planteado.

Para obtener el valor de todos los pesos de los criterios de calidad (W_q), se ha realizado un estudio de cada criterio de forma individual, para así asignar los pesos más importantes a los criterios que mejores resultados aportan a la misión.

Para realizar dichas pruebas, se ha realizado una serie de cálculos variando el valor de los pesos utilizando las tareas propuestas en la tabla 5.1. En este caso, se han calculado los distintos valores de los pesos para los casos de un intervalo de 6 a 12 misiones. Los datos recogidos se pueden ver en la Figura 5.1

N Tareas	Dist	Duracion	Consumo	Media			
6	1	0,5102041	0,8947368	0,735718411	0,83200859	0,812781955	0,79355532
7	1	0,5102041	0,9259259	0,742574265	0,83824641	0,81746032	0,79667423
8	1	1	0,91525424	0,970059881	0,983050848	0,987288136	0,991525424
9	1	1	0,91056913	0,968299721	0,982113826	0,98658537	0,991056913
10	1	0,95238096	0,886076	0,943820246	0,962929488	0,966244736	0,969559984
11	1	0,95238096	0,86868685	0,937159458	0,959451658	0,963636364	0,967821069
12	1	0,95238096	0,86538464	0,935875229	0,958791216	0,963141032	0,967490848
Media	1	0,748129691	0,899794644	0,878891162	0,926652764	0,92213788	0,917288532
A: 50%Dist + 30%Dur + 20%Consumo					Mejor resultado		
B: 50%Dist + 35%Dur + 15%Consumo							
C: 50%Dist + 40%Dur + 10%Consumo							

Figura 5.1: Tabla de datos para cálculo de pesos

Como punto de partida, se ha optado por darle un peso constante a la función de evaluación de distancia. Según las pruebas de ejecución, ha mostrado un mejor comportamiento en términos de estimación y valor de fitness obtenido. Por otro lado, esta función está directamente relacionada con el consumo, por lo que, dando un peso más importante a este criterio, tendrá un efecto positivo en el criterio de consumo.

Como se observa en la Figura 5.1, la combinación de los pesos de cada criterio de calidad $W_{dist} = 0.50$, $W_{dur} = 0.30$ y $W_{fuel} = 0.20$, es la que mejor rendimiento ha obtenido con una media total de 0,926652764. Por ello, el resto de resultados de las siguientes secciones serán obtenidos con los pesos aquí calculados.

5.3. Resultados experimentales

En esta sección se van a mostrar los datos obtenidos mediante la experimentación con el sistema desarrollado según diversos parámetros.

5.3.1. Evaluación de misiones basado en las tareas

En esta sección se mostrarán los datos obtenidos según el número de tareas de la misión evaluada. Las pruebas se realizarán con las tareas especificadas en la tabla 5.1, con todos los

UAVs de la colección. A continuación, se estudiarán los siguientes parámetros:

- **Asignación de tareas**: En este campo se muestra la asignación de tareas que recibe cada UAV que participa en la misión. La nomenclatura es $U_x[T_1..T_n]$ donde x es el identificador del UAV y T_i cualquier tarea de la misión.
- **Valor de fitness** F_{val}: Es la representación del grado de adaptación del cromosoma al problema, o lo que es lo mismo, la eficacia con la que el cromosoma puede resolver la misión establecida.
- **Punto de convergencia** P_{conv}: Este valor determina la generación donde la fitness converge a un valor para la misión dada (puede no existir punto de convergencia).
- **Duración de la misión**: Tiempo total desde el inicio de la primera tarea hasta el final de la última, donde todos los UAVs han completado sus tareas asignadas.
- **Distancia recorrida total**: Distancia total recorrida por todos los UAVs en la misión.
- **Consumo total de combustible**: Total de litros de combustible consumido durante la misión.

N Tareas	Asignación de tareas	F_{val}	P_{Conv}	Duración	Dist	Consumo
1	**U2**[0]	0.964	0	20 min	3.335 Km	10.341 L
2	**U0**[1] **U2**[0]	0.967	0	20 min	5.559 Km	16.568 L
3	**U0**[1] **U2**[0] **U4**[2]	0.971	0	30 min	6.671 Km	18.791 L
4	**U0**[1] **U2**[0] **U4**[2,3]	0.978	1	50 min	10.007 Km	25.463 L
5	**U0**[1] **U2**[0] **U4**[2,3,4]	0.985	1	65 min	15.567 Km	36.583 L
6	**U0**[1,5] **U2**[0] **U4**[2,3,4]	0.981	3	65 min	20.015 Km	49.036 L
7	**U0**[1,4,5] **U2**[0] **U4**[6,2,3]	0.983	6	60 min	23.350 Km	55.708 L
8	**U0**[1,4] **U2**[0,5,7] **U4**[6,2,3]	0.978	8	50 min	30.022 Km	76.724 L
9	**U0**[1,4,5] **U2**[0,7] **U4**[6,2,3] **U5**[8]	0.957	12	120 min	36.694 Km	127.095 L
10	**U0**[1] **U1**[0] **U2**[4,5,7] **U4**[6,2,3] **U5**[8,9]	0.907	34	135 min	36.844 Km	181.803 L
11	**U0**[1,10] **U1**[6,7] **U2**[0,3] **U3**[4,5] **U4**[2] **U5**[8,9]	0.901	35	135 min	36.694 Km	211.715 L
12	**U0**[1,10] **U1**[0,3,11] **U2**[4,5] **U3**[6,7] **U4**[2] **U5**[8,9]	0.899	31	135 min	36.694 Km	227.060 L
13	**U0**[1] **U1**[6,12,7] **U2**[0,3,11] **U3**[4,5] **U4**[2] **U5**[10,8,9]	0.899	67	200 min	37.806 Km	249.076 L
14	**U0**[1,3] **U1**[6,12] **U2**[0,13,11] **U3**[4,5,7] **U4**[2] **U5**[10,8,9]	0.858	60	200 min	36.694 Km	253.859 L
15	**U0**[1,3] **U1**[6,12] **U2**[0,13,11] **U3**[4,5,7,14] **U4**[2] **U5**[10,8,9]	0.858	60	200 mins	37.806 Km	255.859 L

Tabla 5.4: Ejemplo de resultados obtenidos en misiones

A continuación, pasamos a desglosar los datos obtenidos tomando como referencia el número de tareas de las misiones resueltas en la tabla 5.4. Cabe destacar que todas las misiones, que aparecen en la tabla citada anteriormente, son misiones donde el 100 % de las tareas han sido resueltas de manera satisfactoria, puesto que el sistema se ha planteado para considerar soluciones válidas únicamente las misiones viables al 100 %.

El primero de los atributos analizados es la **asignación de las tareas**. Como podemos observar en la figura 5.2 el sistema tiene una tendencia a mantener un reparto equitativo de las tareas. Esto se traduce en una capacidad de realizar las tareas en paralelo, o lo que es lo mismo, conseguir un tiempo de misión global lo más corto posible.

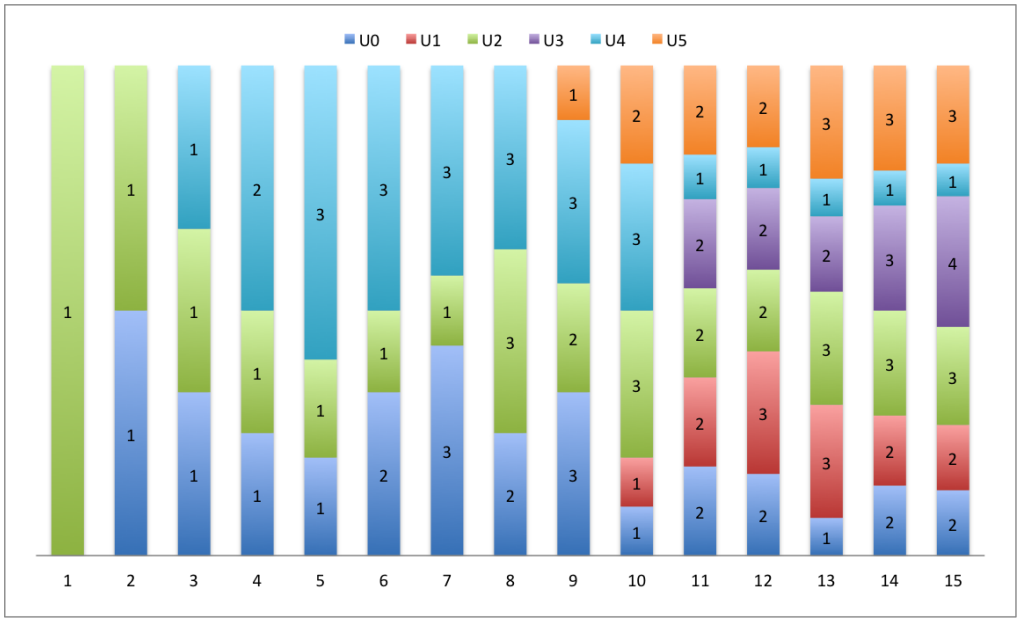

Figura 5.2: Representacion gráfica de la asignación de tareas

Ahora nos ocupamos de forma conjunta de los resultados obtenidos en el valor de **fitness** F_{val} y del **punto de convergencia** de esta P_{conv}. Se puede ver como la función de fitness comienza con valores muy cercanos a 1, es decir, a la solución que previamente se ha estimado como óptima, para ir disminuyendo. En la Figura 5.3 podemos observar como aumenta notablemente (de 0 a 0.6) el número de generaciones necesarias para encontrar una fitness estable, pero el valor de ajuste de esta se mantiene en una variación de un 0.106 puntos.

Resumen total de una misión

En esta sección se muestra el resumen completo que genera el sistema para una misión dada. En este caso, la prueba se ha realizado con la misión formada por las 15 tareas especificadas en la tabla 5.1. Como podemos ver en la Figura 5.4, la salida de datos de la aplicación muestra dos apartados.

Primero se muestra un desglose de cómo han sido asignadas las tareas a los UAVs de la

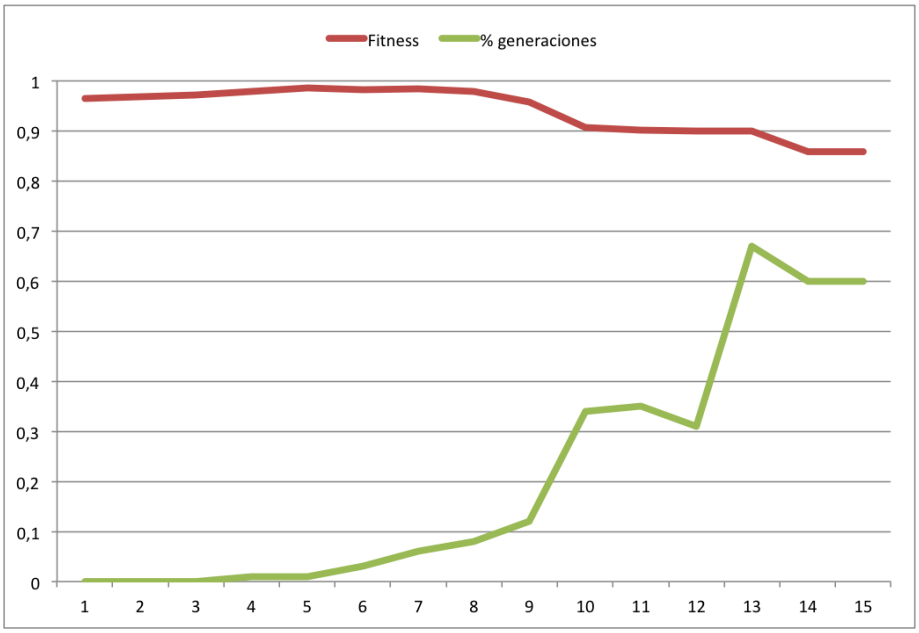

Figura 5.3: Ajuste fitness contra el porcentaje de generaciones de cálculo

colección. Para cada UAV se muestran los datos del conjunto de tareas que ha recibido: la distancia, el consumo y la duración[1].

Después de esto, se muestran los datos totales de la misión. Entre ellos el número total de tareas, junto con el valor de la función de fitness para la solución encontrada. A continuación, se muestra el agregado de todos los atributos antes citados para los UAVs. Y finalmente la hora de inicio y fin de la misión en su totalidad.

```
Asignacion de tareas
UAV0 Tareas: [1, 13]| Dist: 4.447 Km| Consumo: 12.453 l| Duracion: 49 min, 59 sec
Asignacion de tareas
UAV1 Tareas: [6, 12]| Dist: 7.783 Km| Consumo: 50.593 l| Duracion: 72 min, 59 sec
Asignacion de tareas
UAV2 Tareas: [0, 3, 11]| Dist: 4.447 Km| Consumo: 13.788 l| Duracion: 84 min, 59 sec
Asignacion de tareas
UAV3 Tareas: [4, 5, 7, 14]| Dist: 4.447 Km| Consumo: 44.477 l| Duracion: 105 min, 0 sec
Asignacion de tareas
UAV4 Tareas: [2]| Dist: 1.111 Km| Consumo: 2.223 l| Duracion: 9 min, 59 sec
Asignacion de tareas
UAV5 Tareas: [10, 8, 9]| Dist: 15.567 Km| Consumo: 132.321 l| Duracion: 149 min, 59 sec

Resumen de mision
Numero de tareas: 15 Fitness: Fitness: 0.85888916
Distancia total: 37.806 | Consumo total: 255.859 | Duracion: 199 min, 59 sec
Inicio: 01:00:00 CET      Fin: 04:19:59 CET
```

Figura 5.4: Resumen completo de una misión

[1]Por motivos de codificación, la duración de las tareas siempre aparece como la *Duración de la tarea* - 1 min, 59 sec, para evitar problemas de solapamiento de horarios con tareas de con inicio y fin en el mismo horario.

5.3.2. Rendimiento de las misiones

En este apartado nos centraremos en el rendimiento computacional del sistema, evaluando grandes cantidades de tareas. El propósito de esta prueba es estudiar el potencial del sistema para realizar misiones con grandes cantidades de tareas.Debido a esta cantidad, la complejidad se eleva notablemente y, por tanto, se convierte en una tarea complicada para ser realizada de forma manual.

Para la generación de las tareas, se ha optado por seleccionar de forma aleatoria la zona y el sensor que será utilizado en cada una de ellas. Se ha elegido esta opción con el propósito de abarcar el mayor número de posibles tareas y, así conseguir que la simulación se asemeje más a un escenario real. Para realizar este tipo de pruebas, ha sido necesario reconfigurar el comportamiento del algoritmo genético aumentando el número de generaciones que realiza por misión, incrementando este parámetro de 100 a 300. El propósito de este cambio es garantizar un valor de ajuste realmente afinado para el incremento que han sufrido las tareas.

N Tareas	F_{val}	P_{Conv}	Tiempo ejecución
20	0.779	131	5,4 s
40	0.729	220	9,23 s
60	0.714	289	13,63 s
80	0.707	295	17,9 s
100	0.700	284	22,39 s
120	0.691	293	26,69 s
140	0.683	295	31,78 s

Tabla 5.5: Ejemplo de rendimiento en misiones de gran tamaño

En este caso las pruebas estaban centradas en medir el comportamiento del sistema en términos de rendimiento, centrando los resultados en el coste computacional de resolverlas. Por ello, en la tabla 5.5 hemos desechado los datos propios del resultado de la misión y nos centraremos en analizar los siguiente valores:

- **Fitness**: En este caso la función fitness mantiene la tendencia que ya presentaba con en la tabla 5.4. Debido a la gran cantidad de misiones, la fitness se aleja más de su valor óptimo, ya que el incremento de misiones provoca una estimación menos exacta que en casos más controlados, como el que se muestra en la sección 5.3.1.
- **Punto de convergencia**: En esta prueba de rendimiento se observa cómo el punto de convergencia ha incrementado su valor de forma notable, alcanzando casi el valor máximo. Es decir, en este punto y a la vista de los datos de la tabla 5.5, se puede afirmar que para misiones compuestas por más de 60 tareas la convergencia no está asegurada. Este comportamiento se puede observar en la gráfica de la Figura 5.6, donde a partir de 60 tareas el punto de convergencia alcanza una cota cercana a las 300 generaciones (este comportamiento significa que para realizar misiones de más tareas se necesitarán más de 300 generaciones para garantizar una fitness óptima).
- **Tiempo de ejecución**: Este atributo es el que define realmente esta prueba, dado que para las anteriores el tiempo de cómputo era mínimo (entorno a 1 y 2 segundos). Co-

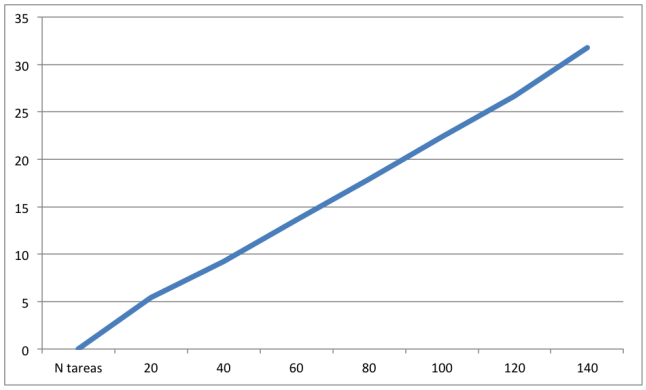

Figura 5.5: Tiempo de cómputo según el número de tareas

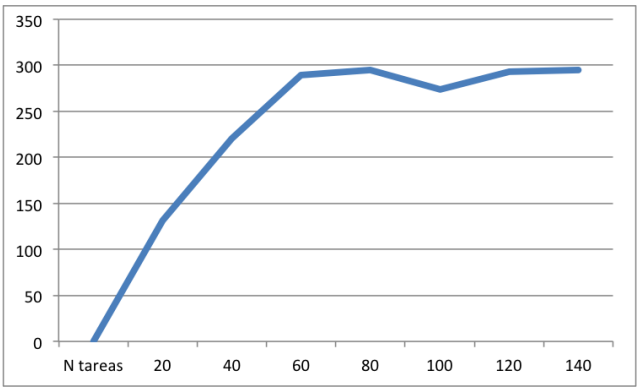

Figura 5.6: Punto de convergencia según el número de tareas

mo se muestra en la Figura 5.5, el tiempo mantiene una tendencia de crecimiento lineal directamente asociada al número de tareas que componen la misión[2].

[2]Los valores de los resultados para el cálculo del rendimiento del sistema se han realizado con un Macbook Pro: 2.5 GHh Intel Core i5, 8GB 1600 MHz DDR3 RAM.

Capítulo 6

Conclusiones

El objetivo principal de este proyecto ha sido la creación de un planificador de misiones para vehículos aéreos no tripulados (UAVs). Con ello se pretende automatizar una tarea que en estos momentos se realiza de forma manual y así abstraer una capa más en la gestión global de las misiones de UAVs.

Para la realización de la planificación, se ha optado por utilizar técnicas basadas en algoritmos genéticos y así ofrecer otro concepto de planificación, alejándose de los planificadores comunes basados en técnicas de búsqueda en grafo. Para realizar esta tarea, se ha profundizado en las características que ofrecen los algoritmos genéticos y los beneficios que estos pueden ofrecer en la planificación de misiones.

Para este proyecto, se ha creado un modelo con los siguientes elementos: **UAVs** (vehículos que realizan las misiones), **Sensores** (elemento que hay que utilizar en cada tarea), **Zonas** (localización de tareas) y **Tareas** (elementos que conforman una misión) a realizar.

Por otro lado, el algoritmo genético ha sido desarrollado mediante la librería de clases Java ECJ y utiliza la estructura de un AG simple ya que cumple los pasos **Evaluación**, **Selección**, **Cruce/Mutación**, **Reproducción**.

Una vez realizados los experimentos, se puede concluir que la utilización de algoritmos genéticos para la planificación es una elección altamente recomendable, ya que consigue resultados potencialmente óptimos en un tiempo de cálculo muy bajo comparado con los tiempos de los algoritmos tradicionales de búsqueda y planificación, dado que, usualmente este tipo de misiones tienden a convertirse en problemas de combinatoria.

Por ello, se puede concluir que para resolver el problema actual de la planificación de misiones para UAVs, especialmente si las misiones que se han de realizar poseen una complejidad elevada debido a la necesidad de coordinar una colección grande de UAVs o a la de resolver un número muy alto de tareas, la utilización de algoritmos genéticos es la más recomendable en términos de eficiencia y funcionalidad.

Capítulo 7

Posibles mejoras

En este capítulo se ofrece una serie de mejoras futuras que incorporar a este proyecto, mejorando los elementos ya existentes o añadiendo posibles módulos con nuevas funcionalidades.

Incremento de atributos

En el sistema actual se tienen en cuenta algunos de los atributos más relevantes tanto en los UAVs como en las tareas, pero estos son limitados si se tiene en cuenta todas las posibles variables que pueden aparecer en una misión real. Por ello, una de las posibles mejoras futuras es el aumento de atributos a considerar a la hora de planificar la misión. A continuación se citan unos posibles ejemplos:

- **Altura**: Tener en cuenta la altura que los UAVs son capaces de alcanzar o la necesidad de restringir accesos por altura y no exclusivamente por zona.
- **Sensor**: Aumentar el número de sensores disponibles, dado que los UAV cada vez pueden equipar mayor número de sensores y con ello realizar nuevas tareas.
- **Autonomía**: Considerar además del gasto de combustible, la autonomía que posee un UAV.
- **Velocidad**: Tener en cuenta la velocidad de los UAV para realizar las misiones y con ello disminuir el tiempo de misión.

Tareas relacionadas

Esta mejora consiste en la posibilidad de crear misiones dependientes unas de otras, es decir, tener la posibilidad de especificar que una tarea se ejecute sólo cuando otra ha sido realizada con éxito. Con esto podemos incrementar la variedad de misiones sin la necesidad de establecer un tiempo determinado para su realización.

Mejoras en la estimación de misiones

Incrementar la exactitud de las estimaciones realizadas para las funciones de ajuste, con el objetivo de ofrecer soluciones lo más cercanas a la solución óptima posible.

Mejoras en el rendimiento

Mejorar las funciones de fitness para optimizar el coste de cómputo de estas y, con ello, liberar de carga computacional al algoritmo para así disminuir el tiempo de ejecución, lo que se traduce en la posibilidad de realizar misiones más complejas.

Preferencias en la ejecución de la misión

Dar la posibilidad al usuario de elegir entre algunos de los siguientes criterios para cubrir necesidades concretas.

- **Velocidad**: El usuario puede elegir priorizar la velocidad a costa de valores como el consumo.
- **Consumo**: El usuario puede elegir dar preferencia a misiones con poco consumo de combustible, penalizando así el tiempo de misión.
- **Número de UAVs**: El usuario puede limitar el uso de ciertos UAVs, para utilizar únicamente un número de ellos.

Simulación de misiones

Para mejorar la visualización de las misiones para el usuario, se podría enlazar el resultado de la misión obtenida a la entrada de un simulador de vuelo y, con ello, obtener una representación visual de la misión que se va a resolver.

Aprendizaje de misiones previas

Implementar un sistema experto capaz recopilar información y almacenar en una base de datos cada misión completada para poder realizar diferentes análisis que mejoren la efectividad de futuras misiones. De esta forma se podrá considerar otro factor dentro de la planificación: el éxito previo que hayan tenido las misiones con las mismas características que la que se esta evaluando.

Glosario

AG: Algoritmo genético

BBDD: Base de Datos

ECJ: Evolutionary Computation Java

COP: Constraint Optimization Problem

CSP: Constraint Statisfaction Problem

GPS: Global Positioning System

NASA: National Aeronautics and Space Administration

NOAA: National Oceanic and Atmospheric Administration

RPV: Remotely Piloted Vehicle

SAR: Synthetic Aperture Radar (Radar de apertura sintética)

STRIPS: Stanford Research Institute Problem Solver

UAS: Unmanned Air System

UAV: Unmanned Air Vehicle

VANT: Vehículo Aereo no tripulado

XML: eXtensible Markup Language

3D: Tres dimensiones

Bibliografía

[1] Norman Delisle and Mayer Schwartz. A programming environment for csp. In *ACM SIGPLAN Notices*, volume 22. ACM, 1987.

[2] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3), 1972.

[3] Erik J. Forsmo, Esten I. Grotli, Thor I. Fossen, and Tor A. Johansen. Optimal search mission with unmanned aerial vehicles using mixed integer linear programming. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE, 2013.

[4] Brian Handwerk. 5 surprising drone uses. `http://news.nationalgeographic.com/news/2013/06/130606-drone-uav-surveillance-unmanned-domicopter-flight-civilian-helicopter/`, 2013.

[5] John R. Koza and James P. Rice. Automatic programming of robots using genetic programming. In *AAAI*, volume 92, 1992.

[6] Steven. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[7] Kim-Fung Man, Kit-Sang Tang, and Sam Kwong. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics*, 43(5), 1996.

[8] Melanie Mitchell. Genetic algorithms: An overview. *Santa Fe Institute*, 1, 1995.

[9] Sankar K. Pal and Paul P. Wang. *Genetic algorithms for pattern recognition*. CRC press, 1996.

[10] Thomas James Gleason Paul Gerin Fahlstrom. *Introduction to UAV Systems*, volume 1 of *4*. Wiley, The British Library, 4 edition, 7 2012.

[11] Arthur Richards, John Bellingham, Michael Tillerson, and Jonathan How. Coordination and control of multiple uavs. In *AIAA guidance, navigation, and control conference, Monterey, CA*, 2002.

[12] Samuel Sánchez Caballero, Miguel Ángel Sellés Cantó, Miguel Ángel Peydró Rasero, and Rafael Plá Ferrando. Optimización de transmisiones de engranajes mediante algoritmos evolutivos. *Area de innovación y desarrollo S.L.*, 2013.